Wind projection basis for real-time animation of trees

Julien Diener, Mathieu Rodriguez, Lionel Baboud, Lionel Reveret

EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Wind projection basis for real-time animation of trees

Julien Diener[1], Mathieu Rodriguez[2,3], Lionel Baboud[1], Lionel Reveret[1]

[1] Laboratoire Jean Kuntzmann, INRIA Rhône-Alpes
[2] Département de mécanique, LadHyX, Ecole Polytechnique-CNRS
[3] UMR547 PIAF, INRA, Université Blaise Pascal

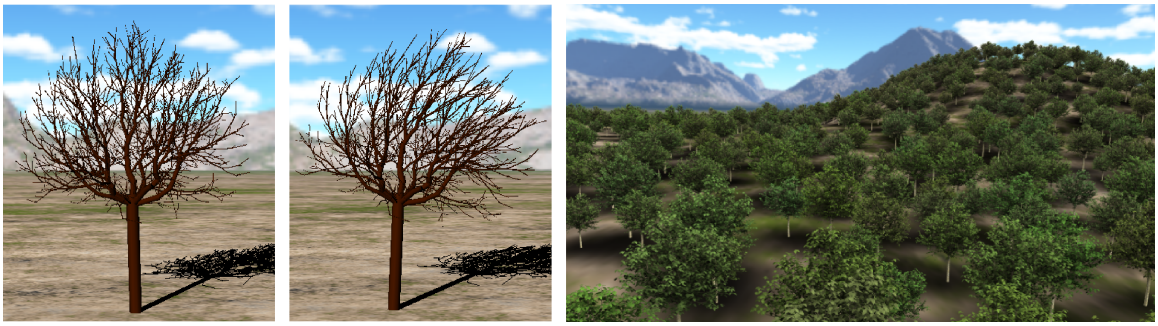

Figure 1: *A single tree and a forest animated with our method.*

Abstract

This paper presents a real-time method to animate complex scenes of thousands of trees under a user-controllable wind load. Firstly, modal analysis is applied to extract the main modes of deformation from the mechanical model of a 3D tree. The novelty of our contribution is to precompute a new basis of the modal stress of the tree under wind load. At runtime, this basis allows to replace the modal projection of the external forces by a direct mapping for any directional wind. We show that this approach can be efficiently implemented on graphics hardware. This modal animation can be simulated at low computation cost even for large scenes containing thousands of trees.

1. Introduction

Vegetation is an important part of the realistic depiction of natural scenes in a virtual environment. While research in plant modeling has produced impressive results, animation techniques remain of visually poor quality, especially for real-time applications.

The current development of computer hardware allows real-time applications to render natural scenes with high quality geometric details. However, the increase in model complexity also generates the need for suitable animation techniques. As the existing accurate simulation methods are still too computationally expensive, developers usually resort to ad-hoc techniques. Such methods can be satisfying for simple models such as grass or palm trees, but fails to provide realistic animation of complex branching structures such as standard trees.

Similarly to [Sta97], we rely on a modal analysis to animate 3D models of trees. This technique simulates the deformation of the plant using a precomputed deformation basis. While it provides visually convincing results, it cannot be used in an interactive application as it requires full integration of the dynamic wind forces over the entire plant at runtime. Real-time framerates can be obtained only for very simple models but not for realistically complex trees.

Our contribution is a method that makes the computation of the wind forces integration almost costless and indepen-

dent of the tree complexity. The only remaining computations at runtime consist in the update of the positions of animation control points, defining the skeleton used to animate the rendered geometry. To this end we define a modal projection basis where the influence of directional wind loads can be precomputed off-line.

After introducing modal animation of trees in section 3, we show in section 4 how the wind projection basis is computed and how it is used to avoid the integration of wind forces at runtime. Finally some implementation issues are discussed in section 5. We show that an entire forest containing thousands of trees can be animated in real time using graphics hardware, producing realistic results for both close and distant trees.

2. State of the art

Many models defined for real-time animation of trees use mechanical simplification specifically designed for that purpose. For example, in [OTF*04] and [AK06] the tree dynamics is modeled using the *static response* of each branch segment (*i.e.* which does not consider time integration of the dynamics). However both methods focus instead on the wind formulation. Ota et al. propose a realistic stochastic description. Akagi et al. incorporate the influence of the plant in the computation of the wind flow modeled with the Navier-Stokes equation.

To animate forests in real-time, Di Giacomo et al. [GCF01] also use static response to animate the tree's terminal branches. For the rest of the structure they simulate the dynamics of the rotation angles between branches using an explicit Euler method, in a similar way as Sakaguchi and Ohya in [SO99].

These previous works model the tree's mechanical response expressed in generalized coordinates. However they solve the dynamics locally (i.e segment by segment) and do not fully model the inertia of the subsequent branches that are displaced by a joint's rotation. Simulation techniques of global mechanical structures such as [Bar96] provide correct dynamics without this limitation. Weber [Web08] developed such a simulation method adapted to trees.

For real-time application only little computation time is commited to the animation of vegetation. The use of parallelizable algorithms has become essential. However animation described by generalized coordinates makes the computation of nodes position hierarchically dependent on each other. Weber proposes a scheme to use multithreading, but it is not yet appropriate for highly parallel computing provided by graphics processing unit (GPU).

Data driven methods are interesting alternatives to direct simulation. Diener et al. [DRF06] use video to extract motion data from real plants and reproduce it on virtual models. James et al. in [JF03] and [JTCW07] use precomputed simulation as input to generate realistic response to external impulse and to fluctuating wind respectively. However the main disadvantage of these approaches is the restricted controls at run time.

First introduced to the computer graphic community by Pentland and Williams [PW89], modal analysis has proven to be a powerful basis for the simulation of oscillatory behaviors. It has then be used for many applications such as precomputed response in an animation using graphics hardware [JP02], real-time collision and manipulation [HSO03] and even sound generation [JBP06].

Modal analysis provides a basis of deformations where all elements of the basis are independent harmonic oscillators. For trees, Shinya and Fournier [SF92] applied this approach to model each of the branches. Stam [Sta97] was the first to use modal decomposition of the entire structure to animate plants. The main idea is to generate a stochastic wind and compute the modal dynamics directly in the frequency domain using the specific response of harmonic oscillators. The goal of this method is to precompute a plant's motion. To use this technique with an interactive wind its modal projection needs to be computed at runtime (see section 4). It would be conceivable on today's graphic hardware but only for coarse tree structures.

Our contribution consists in a method that drastically reduces the computations needed to simulate modal dynamics under controlable wind load. This is done by precomputing a basis of the wind modal projections. Moreover the dynamics is simulated with a computational cost independent of the tree structure complexity and directly proportional to the required quality of the animation. This makes the implementation of level of detail straightforward: the number of animation modes and the subset of tree nodes necessary for rendering can be adjusted freely at runtime.

3. Modal animation framework

In this section, we briefly describe the mechanical model. We introduce modal analysis and explain how it can be used in a real-time animation framework.

Mechanical modeling and modal analysis have been thoroughly studied. We advise interested readers to refer to specialized literature such as [GR94].

3.1. Mechanical model

To solve dynamics over time we resort to numerical simulation applied on a discretized model of a tree obtained using the Finite Element Method (FEM). The principle is to decompose the mechanical system into a set of *elements* separated by *nodes*. The continuous system's deformation is defined as a specific interpolation between nodes displacements.

In our case, as in [Sta97], the system domain Ω is the *lineic* representation of the tree, i.e. the set of central axes

of the branches. Then the FEM discretizes Ω as a set of segments Ω^e (see figure 2).

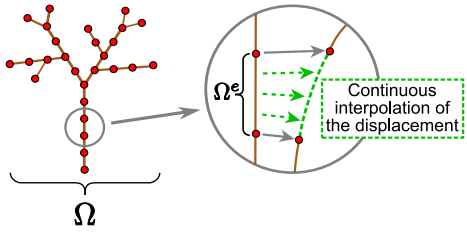

Figure 2: *Finite element discretization.*

Let $\mathbf{u}(t)$ be the vector containing the displacements of all the nodes at time t. The dynamics of a dissipative system is commonly expressed by the equation of motion:

$$M\ddot{\mathbf{u}}(t) + C\dot{\mathbf{u}}(t) + K\mathbf{u}(t) = \mathbf{f}(t) \tag{1}$$

where $\mathbf{f}(t)$ is the external force (see section 4), M and K are the mass and stiffness matrices. The commonly accepted model for the damping matrix C is a linear combination of M and K :

$$C = \alpha M + \beta K \tag{2}$$

More details on the finite element representation and the matrices used in equation (1) are given in appendix.

3.2. Modal analysis

Modal analysis is the decomposition of the deformations of a mechanical system into a set of special deformations called *vibration modes*. These modes are the solutions $\mathbf{u}(t)$ of the free vibration equations:

$$M\ddot{\mathbf{u}}(t) + K\mathbf{u}(t) = 0 \tag{3}$$

$$\mathbf{u}(t) = \lambda_i(t) \cdot \varphi_i \tag{4}$$

where the φ_i are constant vectors called the *modal deformations*, and the $\lambda_i(t)$ are scalar functions of time. Substituting (4) in (3) gives:

$$M^{-1}K\varphi_\mathbf{i} = \mu_i\varphi_\mathbf{i} \quad \text{with} \quad \mu_i = -\frac{\ddot{\lambda}_i(t)}{\lambda_i(t)} \tag{5}$$

As K, M and φ_i are constants, μ_i are constant scalars.

Finally, the eigenvectors and eigenvalues of $M^{-1}K$ are the solutions of equation (5). Let Φ be the matrix containing all the modal deformations φ_i, and $\mathbf{q}(t)$ the expression of vector $\mathbf{u}(t)$ in eigenspace:

$$\Phi\mathbf{q}(t) = \mathbf{u}(t) \tag{6}$$

Substituting (6) in equation (1) and multiplying on the left by Φ^T, we get:

$$\underline{M}\ddot{\mathbf{q}}(t) + \underline{C}\dot{\mathbf{q}}(t) + \underline{K}\mathbf{q}(t) = \Phi^T\mathbf{f}(t) \tag{7}$$

where $\underline{M} = \Phi^T M\Phi$, $\underline{C} = \Phi^T C\Phi$ and $\underline{K} = \Phi^T K\Phi$. A classical result of modal analysis is that these matrices are diagonal [GR94]. We can then rewrite equation (7) as $6n$ independent equations such that, for each element q_i of vector $\mathbf{q}$:

$$\ddot{q}_i(t) + \gamma_i\dot{q}_i(t) + \omega_i^2 q_i(t) = \frac{1}{m_i}f_i(t) \tag{8}$$

$$\text{with} \quad \begin{cases} \omega_i^2 = \dfrac{k_i}{m_i} \\ \gamma_i \ = \alpha + \beta * \omega_i^2 \end{cases} \tag{9}$$

where $f_i(t)$ is the *modal force* induced by the wind (see section 4), ω_i the natural frequency, γ_i the damping coefficient of mode i, m_i and k_i the modal mass and damping respectively (i.e the diagonal elements of $\underline{M}$ and $\underline{K}$).

3.3. Animation using the modes of deformation

Modal analysis is well suited to our purpose because it extracts the most representative components of the structure deformations. For the case of plants, the vibration modes associated with low frequencies (ω_i) have the most significant (and largely distributed) influence on the final motion [RdLM08]. Typically the first modes (of lower frequencies) are large deformations of the whole tree. Then as the modal frequencies increase, the deformation becomes more and more confined on the last tiny branches (see figure 4). Hence only a small subset of the mode matrix Φ needs to be kept in order to model most of the motion complexity.

Equation (8) means that each mode behaves as a one-dimensional harmonic oscillator (see figure 3). Each vibration mode can be seen as a deformation of the whole structure oscillating independently from the others.

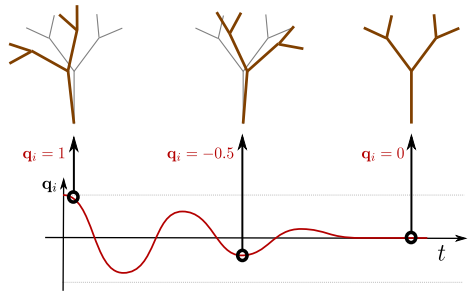

Figure 3: *A vibration mode is a deformation of the whole tree behaving as a harmonic oscillator.*

At each time step, the simulation of modal animation consists in the following procedure:

1. Compute $f_i(t)$, the modal projection of the wind.
2. Update the dynamics of the modes using the explicit Euler method:

$$\begin{aligned} \ddot{q}_i^{t+\delta t} &\leftarrow f_i(t)/m_i - \gamma_i\dot{q}_i^t - \omega_i^2 q_i^t \\ \dot{q}_i^{t+\delta t} &\leftarrow \dot{q}_i^t + \delta t\ \ddot{q}_i^{t+\delta t} \\ q_i^{t+\delta t} &\leftarrow q_i^t + \delta t\ \dot{q}_i^{t+\delta t} \end{aligned}$$

3. Compute the updated displacement:

$$\mathbf{u}^{t+\delta t} = \sum_{\forall i} q_i^{t+\delta t} \varphi_i \tag{10}$$

Finally, only the evaluation of the modal forces $f_i(t)$ requires costly computations. This is the main difficulty that our technique proposes to resolve. To this end we now describe a basis that can be computed off-line and avoids costly computations at runtime.

4. Wind projection basis

Integrating a general wind load over the whole tree requires a sum over all the elements (branches) at each time step of the animation (as done by [Sta97]). Doing so prevents real-time animation of sufficiently complex trees. By making the assumption that the wind load is uniform over the whole tree, a projection basis can be precomputed that allows to animate the tree in real-time under the load of any directional wind.

Note that the wind is considered uniform only over each tree but can vary between distinct instances. The speed and direction of the wind are left unconstrained and can be freely controlled at runtime.

We start by modeling the drag load induced by the contact of wind on a rigid structure. As in [SFL06, dL08], we model the wind load on any point $p \in \Omega$ at time t by:

$$\mathbf{f}(p,t) = \frac{1}{2}\rho C_D D(p) \|\mathbf{v}(p,t) - \dot{p}\| (\mathbf{v}(p,t) - \dot{p}) \tag{11}$$

where $\mathbf{v}(p,t)$ is the wind speed vector and $D(p)$ is the diameter of the branch at point p. The air density ρ and the drag coefficient C_D, depending on the shape of the branches, are constant. We merge them as one scalar factor C.

From this local definition, one can derive the modal wind load for each mode i by modal projection [GR94]:

$$f_i = \int_\Omega \mathbf{f}(p) \cdot \varphi_i(p) dp \tag{12}$$

where the time parameter is omitted for clarity and $\varphi_i(p)$ is the vector containing the i^{th} modal deformation at p.

We do not take into account the influence of leaves and the direction of branches in equation (11). The drag of a single leaf is a complex phenomenon still not well understood. We observed that taking it into account using an isotropic drag model (that can be seen as a statistical mean) does not does not have a significant influence on the modal decomposition. Thus we decided to omit it in our computations. For branches, wind forces should be projected taking into account branches directions. Because the modes with low frequency are mostly perpendicular to the branches, the modal projection (the dot product of $\mathbf{f}$ and φ in equation (12)) has a similar effect.

In the case of a freely defined finite element discretization, equation (12) is computed as the sum of the elements contribution:

$$f_i = \sum_e \int_{\Omega_e} \mathbf{f}(p) \cdot R^e N^e(p) \varphi_i^e dp \tag{13}$$

where R^e is the rotation matrix mapping the local frame of the element e to the global frame. φ_i^e is the vector containing the i^{th} modal deformation of both element's border nodes (in the local frame) and $N^e(p)$ is the *shape function matrix* defining the interpolation between these nodes (see appendix).

To compute the modal forces, equation (12) can be discretized over the nodes of the FEM (in the frequency domain) as in [Sta97]:

$$\widehat{f}_i = \sum_{n \in \Omega} \widehat{f}(n) \cdot \varphi_i(n)$$

For simple structure it is a suitable approach but for complex trees such a modal projection it would be prohibitive. Using a simplified structure is not a solution as it would result in a poor approximation of the effective modal stress.

In our method, to reduce the cost of the modal projection of the wind forces at runtime, we focus on extracting the time dependent parameters out of the integral. The remaining part is then precomputed.

The first simplification is to consider R^e as a constant: the wind load is computed on the tree in its rest position. The orientation changes of the branches is not significant as they remain small for natural trees motion. Compensatory models were tested but did not bring noticeable improvement. With this approximation, the modal deformation term is constant and only the wind force depends on time. To extract runtime parameters, we assume a wind model such that:

1. From the wind force, we only keep the *mean aerodynamic stress*:

$$\mathbf{f}(p) = C\, D(p) \|\mathbf{v}(p,t)\| \mathbf{v}(p,t)$$

   The remaining part of the external force, called the *aerodynamic damping*, can be compensated by an increase of the damping coefficients α and β from equation (2).
2. The wind has a spatially uniform speed $\mathbf{v}(t)$ over a single tree.

Using these simplifications, equation (12) becomes:

$$f_i = \int_\Omega C\, D(p) \|\mathbf{v}(t)\| \mathbf{v}(t) . \phi_i(p) dp \tag{14}$$

$$= C \|\mathbf{v}(t)\| \mathbf{v}(t) . \int_\Omega D(p) \phi_i(p) dp \tag{15}$$

It results that if we precompute the wind projection vectors p_i defined as:

$$p_i = C \int_\Omega D(p) \phi_i(p) dp$$

then the modal wind load can be easily computed at runtime using:

$$f_i = \|\mathbf{v}(t)\| \mathbf{v}(t) . p_i \tag{16}$$

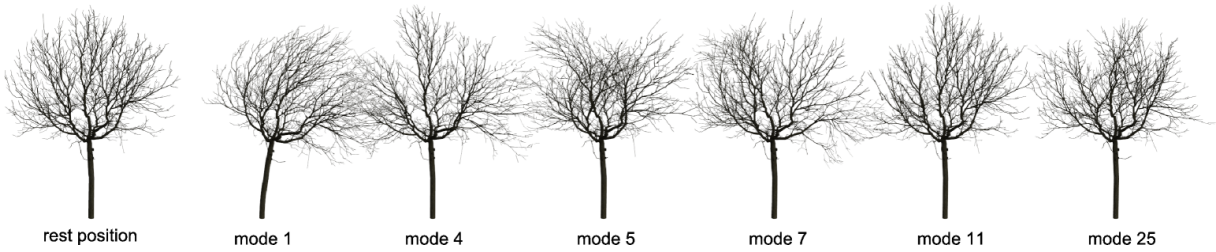

Figure 4: *The modes of deformation for the walnut model. During animation the final displacement of a tree is a combination of its modal deformations. The modes are sorted in ascending order of their natural frequencies.*

5. Implementation issues

Mechanical modeling, modal analysis and precomputation of the projection basis require the use of complex algorithms that must be performed using specialized tools. We used the software Cast3m 2000 [CAS] specialized in the study of mechanical structures using the finite element method.

5.1. GPU implementation

A powerful aspect of the modal representation is that most computations are completely independent. It thus allows to take full advantage of the massive parallel computing power offered by present graphics hardware (GPU). This enables us to animate large forest scenes in real time (see section 6).

The two main parameters of our tree model are the number of nodes of the skeleton and the number of modes kept for animation.

1. As can be seen in equation (10) the updated position of each node can be computed independently from the others using only the dynamic modal state (q_i). Thus the nodes positions can be stored in a texture in GPU memory (matrix $\mathcal{U}$ of figure 5) and updated using a fragment program and an offscreen buffer. Animating the tree mesh can then be done by reading this texture in a dedicated vertex program. Any level-of-detail rendering model using a (possibly time-varying) subset of the animation nodes can be used, without the need to update the remaining unused nodes.
2. Adjusting the number of modes used for the computation of updated nodes positions gives a handful tradeoff between animation quality and framerate. For distant views of forests only the first few modes needs to be kept, while adding more modes at close range adds detail to the animation. As for the nodes, each modal state $(q_i, \dot{q}_i)$ can be updated independently from the others, they are stored in a texture in GPU memory (matrix $\mathcal{Q}$ of figure 5), and updated by a dedicated fragment shader.

The overall animation process can be summarized in three main steps (see figure 5).

1. Update of the modal state for each mode of each tree (item 1 and 2 of section 3.3). It requires to compute the local wind speed for each tree instance.
2. Deformation of the skeletons from the rest position (item 3 of section 3.3)
3. Rendering each tree with deformed geometry using skinning on the animated skeleton.

5.2. Error correction

Modal deformation being a linear approximation of the displacement of the branches nodes, it is mostly adapted for low amplitude motion. Notably, branches lengths are not kept constant when submitted to strong wind. We cannot rely on using rotating branches (like in [Zio07]) as it would break parallelism between nodes computations and does not extend to complex structures. This classical issue of modal analysis could be addressed by modal warping [CK05]. In our specific case of tree animation we observed that when a single mode i is selected, the corrected displacement of each node n lies on a smooth trajectory (see figure 6) which can be faithfully approximated by a quadratic curve $\mathbf{u}$ after a non linear reparameterization s:

$$\mathbf{u}(q) = s \cdot \mathbf{v} + s^2 \cdot \mathbf{w} \quad \text{with} \quad s = a \arctan(b \cdot q) \tag{17}$$

Our error correction consists in replacing equation (10) by:

$$\mathbf{u}_n = \sum_i s_{i,n} \cdot \mathbf{v}_{i,n} + (s_{i,n})^2 \cdot \mathbf{w}_{i,n} \tag{18}$$

$$\text{with} \quad s_{i,n} = a_{i,n} \arctan(b_{i,n} \cdot q_i) \tag{19}$$

Vectors $\mathbf{v}_{i,n}$, $\mathbf{w}_{i,n}$ and scalars $a_{i,n}$, $b_{i,n}$ are stored in matrix Φ (figure 5) instead of the displacement vectors (roughly doubling its size). They are optimized using uniformly sampled deformations, corrected by pulling each node towards its parent to match initial branches lengths, in a depth-first fashion. The results show that this approach yields a sufficiently good correction for branches lengths to allow the application of a strong wind load.

6. Results

We tested our technique on various models on an Intel Xeon 3.2 GHz with a GeForce 8800 GTS. The following results

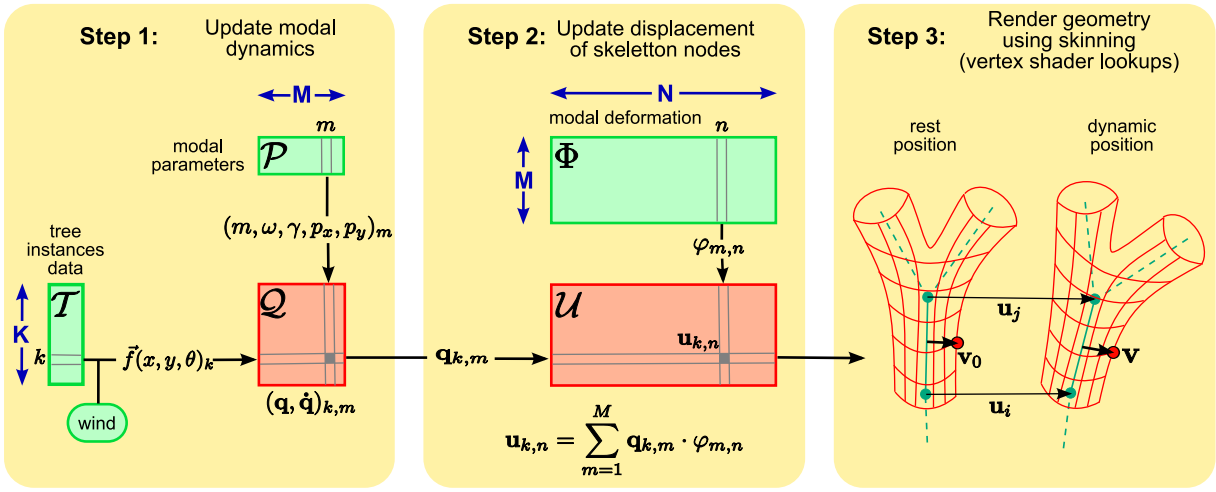

Figure 5: *Computation steps of the animation framework. K is the number of tree instances, M the number of modes used for the animation and N the number of control points of the skeleton. The colored rectangles represent textures stored in GPU memory. The input matrices $\mathcal{T}$, $\mathcal{P}$, Φ are constant while the matrices $\mathcal{Q}$, $\mathcal{U}$ store the dynamic variables. $\mathcal{T}$ contains the tree instance data (position, orientation); $\mathcal{P}$ contains the modal parameters used to compute the dynamics; Φ contains the modal deformations of the skeleton control points; $\mathcal{Q}$ contains the modal states $(q_i, \dot{q}_i)$; $\mathcal{U}$ contains the skeleton's node displacements. Several representations can be used for the wind such as a procedural equation or an animated flow field.*

Figure 6: *Correct trajectories obtained by selecting each mode successively (one color per mode). These trajectories can be faithfully approximated by second degree polynomial curves.*

were obtained on a walnut tree digitized from precise measurements on a real tree (from [SRG97], see figure 8), with 25 modes. The accompanying video involves this model and an oak model with 50 modes generated using [RLP07] (see figure 7). It demonstrates the resulting realism and some of the possibilities offered by the presented technique.

Figure 7: *The oak tree model.*

Our implementation can animate and render over 4000 trees while maintaining real-time framerates (30 Hz minimum). For the walnut tree with 3437 nodes, the undecimated version of the mesh is made of approximately 120000 vertices while the most decimated level of details only uses 300 vertices. Note that our LOD implementation is unsophisticated and could be much more optimized. The only significative data stored in GPU memory is the nodes displacements texture (array $\mathcal{U}$ in figure 5) whose size is $K \times N$ (*e.g.* 10 Mb for 1000 trees with 3000 control points each).

Tables 1 and 2 show that it is always possible to reach high framerates by making a tradeoff between the number of trees and the animation level of detail (*i.e.* the number of nodes and the number of modes).

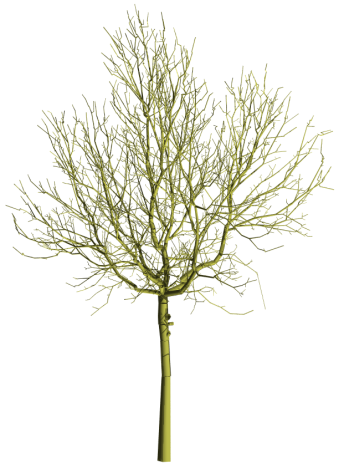

Figure 8: *Digitalized walnut tree.*

K \ N	20	100	500	1000	2000	8000
100				712	385	103
1000		690	169	87	43	11
8000	426	102	21	11		

Table 1: *InFluence of the number of trees (K) and the number of nodes (N) on animation framerates (in Hz), with 25 modes and rendering deactivated.*

M \ K	500	1000	2000	4000
5	73	39	19	10
15	57	29	14	7
25	43	22	11	6

Table 2: *Influence of the number of modes (M) and the number of trees (K) on the framerate (in Hz). The model has 3437 nodes and approximately twice as much vertices, and the LOD system is deactivated to ease interpretation (*i.e. *every node is always updated).*

7. Limitations and future work

As explained in section 3.3, the omission of the vibration modes with highest frequencies strongly reduces the computational cost of simulation but also removes the high frequency motion of tiny branches and leaves. However this can be replaced by using techniques such as [OTF*04] for close view and animated textures for distant views. In our implementation, the leaves rigidly follow the nodes they are attached to. But the complexity of branches animation already provides a very convincing motion.

Mathematically, the main limitation of our method is the assumption that the wind is spatially uniform over each tree. In particular the attenuation of the wind by the tree is not taken into account (*i.e.* the branches in the 'back' of the tree should receive less wind). It does not lower visual realism as the modal dynamics keeps natural oscillatory behaviours. However precomputing a more expressive basis would allow to increase animation precision.

Finally the parallelism of the method coupled with efficient GPU programming enables the animation of thousands of trees. A more aggressive level-of-detail scheme would allow real-time rendering of even larger forest scenes.

8. Conclusion and acknowledgement

We showed a new approach to compute the modal projection of the wind load allowing a drastic reduction of computations at runtime. To this end, we introduce a precomputed basis for the projections of interactive directional wind in a modal animation framework.

Our implementation shows that it is possible to efficiently animate and render thousands of trees. We think the presented technique is perfectly adapted to real-time applications such as computer games or simulators.

This work has been supported by the ANR grant Chêne-Roseau. We would like to thank Adam Runions for providing us many high quality tree models.

Appendix

In the lineic representation of the tree, each element is defined in its local frame where the x-axis is in the direction of the segment at rest. The deformation of an element is the interpolation of its border nodes displacement: linear for the main axis and Hermitian for the y and z axis.

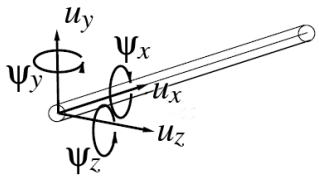

Figure 9: *The six degrees of freedom of a node (figure taken from [Sta97]).*

A node has thus six degrees of freedom (see fig. 9): the three spatial displacements (u_x, u_y, u_z), the torsion around the main axis (ψ_x) and the derivatives ψ_y and ψ_z defined by [GR94]:

$$\psi_y = -\frac{du_z}{dx} \quad \text{and} \quad \psi_z = \frac{du_y}{dx}$$

This interpolation is defined such that, at time t, the displacement $\delta_p(t)$ of any point $p \in \Omega^e$ is:

$$\delta_p(t) = N^e(p)\mathbf{u}^e(t) \tag{20}$$

where $\mathbf{u}^e(t)$ is the 12-dimensional vector containing the displacements of the two element's border nodes, and $N(p)$ the *shape function matrix* is:

$$N^e(p) = \begin{pmatrix} l_1 & . & . & . & . & . & l_2 & . & . & . & . & . \\ . & h_1 & . & . & . & h_2 & . & h_3 & . & . & . & h_4 \\ . & . & h_1 & . & h_2 & . & . & . & h_3 & . & h_4 & . \\ . & . & . & l_1 & . & . & . & . & . & l_2 & . & . \end{pmatrix}$$

where the dots represent zeros, l_1 and l_2 are the coefficients of a linear interpolation and the h_i are the cubic Hermite basis functions.

In accordance with most models of trees, branches are modeled as cylindrical beams [SFL06]. The local stiffness K^e and mass M^e matrix are defined as the discretization of the potential energy ν and kinetic energy τ of an element of length l [GR94]:

$$\begin{aligned} \nu &= \frac{1}{2}\int_0^l EA(\delta u_x)^2 + EI((\delta^2 u_y)^2 + (\delta^2 u_z)^2) + GJ(\delta\psi_x)^2 dx \\ &= \frac{1}{2}\mathbf{u}^{eT} K^e \mathbf{u}^e \end{aligned}$$

$$\begin{aligned} \tau &= \frac{1}{2}\int_0^l \rho A\left(\dot{u_x}^2 + \dot{u_y}^2 + \dot{u_z}^2\right) + \rho J\,\dot{\psi_x}^2 dx \\ &= \frac{1}{2}\dot{\mathbf{u}}^{eT} M^e \dot{\mathbf{u}}^e \end{aligned}$$

Here the dot notation refers to time derivatives and $\delta = \frac{d}{dx}$.

In our implementation, we typically use the following values: $E = 10^{10} Pa$, $G = 2.6E$, $\rho = 10^3 kg.m^{-3}$. From the beam model used, we have $I = \pi r^4/16$, $J = \sqrt{2}I$, $A = \pi r^2$ where r is the average radius of the element.

These local matrices are then transformed to express the energies in function of the nodes displacements defined in the global frame of the tree. Finally, they are assembled together in the global matrices M and K of equation (1). A description of this procedure has been given by [Sta97] or can be found in specialized literature such as [GR94].

References

[AK06] AKAGI Y., KITAJIMA K.: Computer animation of swaying trees based on physical simulation. *Computers & Graphics 30*, 4 (2006), 529–539.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 137–146.

[CAS] Cast3m 2000. http://www-cast3m.cea.fr/.

[CK05] CHOI M. G., KO H.-S.: Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 91–101.

[dL08] DE LANGRE E.: Effects of wind on plants. *Annual Review of Fluid Mechanics 40* (2008), 141–168.

[DRF06] DIENER J., REVÉRET L., FIUME E.: Hierarchical retargetting of 2D motion fields to the animation of 3D plant models. In *Symposium on Computer Animation, SCA 06, September, 2006* (Sept. 2006), ACM-Siggraph/Eurographics, pp. 187–195.

[GCF01] GIACOMO T. D., CAPO S., FAURE F.: An interactive forest. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (2001), Springer, pp. 65–74.

[GR94] GÉRADIN M., RIXEN D.: *Mechanical Vibrations: Theory and Application to Structural Dynamics*. 1994.

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. In *Graphics Interface* (2003), pp. 247–256.

[JBP06] JAMES D. L., BARBIČ J., PAI D. K.: Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3 (Aug. 2006).

[JF03] JAMES D. L., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3 (July 2003), 879–887.

[JP02] JAMES D. L., PAI D. K.: Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 582–585.

[JTCW07] JAMES D. L., TWIGG C. D., COVE A., WANG R. Y.: Mesh ensemble motion graphs: Data-driven mesh animation with constraints. *ACM Transansaction Graphics 26*, 4 (2007).

[OTF*04] OTA S., TAMURA M., FUJIMOTO T., MURAOKA K., CHIBA N.: A hybrid method for real-time animation of trees swaying in wind fields. *The Visual Computer 20* (2004), 613–623(11).

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: modal dynamics for graphics and animation. *ACM Transactions on Graphics, Proceedings of the SIGGRAPH conference 23*, 3 (1989), 207–214.

[RdLM08] RODRIGUEZ M., DE LANGRE E., MOULIA B.: A scaling law for the effects of architecture and allometry on tree vibration modes. *Accepted in American Journal of Botany* (2008).

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling Trees with a Space Colonization Algorithm. Ebert D., Merillou S., (Eds.), Eurographics Association, pp. 63–70.

[SF92] SHINYA M., FOURNIER A.: Stochastic motion-motion under the influence of wind. *Comput. Graph. Forum 11*, 3 (1992), 119–128.

[SFL06] SELLIER D., FOURCAUD T., LAC P.: A finite element model for investigating effects of aerial architecture on tree oscillations. *Tree Physiology 26* (2006), 799–806.

[SO99] SAKAGUCHI T., OHYA J.: Modeling and animation of botanical trees for interactive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '99)* (1999), pp. 139–146.

[SRG97] SINOQUET H., RIVET P., GODIN C.: Assessment of the three-dimensional architecture of walnut trees using digitising. *Silva Fennica 31* (1997), 265–273.

[Sta97] STAM J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Comput. Graph. Forum 16*, 3 (1997), 159–164.

[Web08] WEBER J. P.: Fast simulation of realistic trees. *IEEE Computer Graphics and Applications 28*, 3 (2008), 67–75.

[Zio07] ZIOMA R.: *Gpu gems 3*. Addison-Wesley Professional, 2007, ch. Gpu-generated procedural wind animations for trees (chapter 6).